Pers Ubiquit Comput (2016) 20:283–293
DOI 10.1007/s00779-016-0917-9

ORIGINAL ARTICLE

An efficient algorithm for partially matched services in internet of services

Mariwan Ahmed[1] · Lu Liu[1] · James Hardy[1] · Bo Yuan[1,2] · Nick Antonopoulos[1]

Received: 10 October 2015 / Accepted: 30 March 2016 / Published online: 11 May 2016

Abstract Internet of Things (IoT) connects billions of devices in an Internet-like structure. Each device encapsulated as a real-world service which provides functionality and exchanges information with other devices. This large-scale information exchange results in new interactions between things and people. Unlike traditional web services, internet of services is highly dynamic and continuously changing due to constant degrade, vanish and possibly reappear of the devices, this opens a new challenge in the process of resource discovery and selection. In response to increasing numbers of services in the discovery and selection process, there is a corresponding increase in number of service consumers and consequent diversity of quality of service (QoS) available. Increase in both sides' leads to the diversity in the demand and supply of services, which would result in the partial match of the requirements and offers. This paper proposed an IoT service ranking and selection algorithm by considering multiple QoS requirements and allowing partially matched services to be counted as a candidate for the selection process. One of the applications of IoT sensory data that attracts many researchers is transportation especially emergency and accident services which is used as a case study in this paper. Experimental results from real-world services showed that the proposed method achieved significant improvement in the accuracy and performance in the selection process.

Keywords Emergency service selection · Quality of Service (QoS) · Internet of Things (IoS) · Emergency service ranking · Partial matching

✉ Lu Liu
L.Liu@derby.ac.uk

[1] Department of Computing and Mathematics, University of Derby, Derby DE22 1GB, UK

[2] The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China

1 Introduction

1.1 Overview

With the increasing popularity of Internet of Things (IoT) hardware becoming smaller, cheaper, and more powerful, however, majority of them have computation and communication capabilities which they use to connect, interact and exchange information with surrounding environments [1]. The proliferation of wireless systems such as Bluetooth, Radio Frequency Identification (RFID), Wi-Fi, telephone data services, embedded sensors and actuator nodes has allowed the IoT to develop from infancy, and it is on the verge to transform current static internet to a fully integrated Future Internet [2]. The interaction between all these devices will lead to a large amount of data which needs storing, processing, analysing, and presenting in an efficient, convenient and useable format. In the IoT environment, dynamic network query, discovery, selection, and on-demand provisioning of services are of crucial importance.

One application area of IoT is the emergency and accident management services. The number of emergency cases increases with increasing population and increasing hazard potential from, e.g. the number of cars on the roads. According to Health and Social Care Information Centre (HSCIC) [3], the number of accident and emergency

attendances in England for 2012–2013 is more than 18.3 million. To provide the most suitable service for emergency cases and achieve better performance, Emergency Management Services can provide a unified platform to connect all local and private sector emergency command centres. Concurrently, wireless sensor networks can be used for surveillance and precise automated data collection regarding the emergency case and required services.

There are varieties of attributes which determine the type of emergency service available for a given case. The data are generally dynamic in nature such as crew members, and therefore capabilities, in particular vehicles on a particular shift. While the majority of the data is dynamic, the change rate is not synchronized. For example, some of the data are slowly changing, e.g. vehicles owned by a county service; some are moderately changing, e.g. qualifications and capabilities of crew members and some is rapidly changing e.g. current location and status/availability. To search for the most suitable service, consideration needs to be given to the different service types: fire, police, ambulance, coast guard, anti-terrorist unit, etc. and then the attributes which would include vehicle type (cycle, motorcycle, car, transit vehicle, aircraft), vehicle capability (four wheel drive, number of seats, number of beds), vehicle equipment (defibrillator, oxygen, breathing apparatus, underwater search equipment, access equipment), personnel capability (fully qualified/part qualified/undertaking training in e.g. resuscitation, working at altitude, crash investigation), current service location, current service availability, owner of service provision (county, borough, private, etc.).

Different consumers have different QoS requirements. The diversity in many cases leads to partial matching in which we cannot find enough services which perfect match all the criteria. An example could be for a motorway vehicle accident involving chemical transportation (injuries, vehicle instability, hazardous materials, volatile air supply, chemical clean up, distance to services etc.). In this example a service was not specified, only a partial attribute list. The search would need to be automated based on a reported incident and call handler recording.

For the emergency support service existing technologies in service-oriented systems may be leveraged to provide partial solutions. Within service-oriented architecture (SOA) services are considered as self-contained, self-describing, modular applications that can be published, found, and invoked across the web [4]. SOA stores all the services in repositories, those services can be selected and invoked automatically. Therefore, it can be suggested that it needs more effort to build a more accurate and efficient service selection method to overcome the challenges facing by the process of emergency support service.

In this paper we propose an efficient and dynamic algorithm for selection of disaster services based on partial matching of service QoS attributes. Furthermore, an accurate ranking algorithm is provided by considering the deviation of QoS values from disaster nominal requirements.

From the mentioned points, we can identify three crucial issues as challenges in disaster service selection process:

- Partial matching—where the available emergency services do not fully match every QoS requirements. The degree of partial matching is dependent on the number of recommended services the user needs. In extreme circumstances, this allows our algorithm to recommend the most suitable services instead of returning an empty list. The selection algorithm should avoid excluding those partially matched disaster services.
- Scalability—the algorithm should consider scalability techniques in order to manage large number of emergency services and diversity in the kind of requirements.
- QoS requirements—it is an important part of the selection process, it should be taken into account precisely; the selection process should reflect those requirements for different kinds of emergency services.

1.2 Contributions

Based on the listed challenges of partially matching, scalability, and consumer preferences contributions in this paper can be concluded as;

- A service selection algorithm has been proposed that provides emergency cases a suitable service which matches their requirements.
- The algorithm can achieve a high accuracy since it considers all the available services in the selection process. In the case of not finding an exact match, the algorithm considers partial matching of services.
- The system is able to support a large number of services, and by providing distance correlation weighting mechanism it can support different QoS requirements.
- Utilizing real-world services, the experimental results show that the proposed algorithm improves the quality and performance of the selection process.

The rest of the paper is structured as follows: Sect. 2 shows and analyses some related work in the field of service selection. Section 3 illustrates system architecture and provides detail of ranking algorithm and recommendation of services to the consumer. Section 4 provides experimental results of the work. Finally, Sect. 5 presents the conclusion and draws directions to our future work.

2 Related work

Currently we are witnessing an increase in the number of available services in the IoT environment; this increase requires an automatic and scalable approach for discovery and selection process. There are number of approaches for discovery and selection of services in IoT environment [1, 5–7]. However, discovery and selection is a challenging process especially when there is a list of available services with similar functionality but different QoS parameters such as execution time, security, and cost in line with the increasing number of available services. Many approaches concentrated on discovery or functional matchmaking, for example the studies [8–11] Concentrated on semantic description of services in the form of Input/output and neglected QoS parameters of services; this resulted in a good performance, while accuracy of the selection process is inadequate.

A number of approaches for selection and ranking techniques have been proposed, which take QoS parameters into consideration during discovery and selection as in the papers of Chao et al. [12], Huang et al. [13], Ngan et al. [14] and Kiritikos et al. [15]. With the diversity of QoS parameters, each service consumer wants to select a service based on QoS preferences. Different consumers have different QoS preferences. For example a consumer for flight service may be more interested in price and safety, while a consumer for printing services would be more interested in colour matching and speed of printing.

In a similar approach [16], average ranking was used which treats consumer QoS preferences equally. This negates the priority of different consumers in defining QoS properties. In a paper by Estrella et al. [17], WSARCH was used. WSARCH is based on service-oriented architecture that provides services with verifiable QoS attributes. The approach monitors service providers and analyses data in order to provide suitable services to the consumer's QoS request. In Kritikos and Plexousakis [18], two non-functional matchmaking techniques were proposed. The first one relies on exploiting similarity relationships between offers in order to organize them, while the second one relies on organizing service offers based on non-functional metrics.

The mentioned approaches consider QoS parameters to be precise and be provided by the consumer. But in general QoS parameters from consumers are imprecise and expressed in different forms. In addition little or no detail is given on how to calculate the dependencies between QoS parameters, which will have influence on the service ranking and selection. Fuzzy logic has been used to deal with ambiguous QoS parameters [19–21]. Used fuzzy logic to deal with QoS parameters, however, the approaches provide solution to ambiguous and imprecise QoS parameters and did not address any dependencies between QoS parameters.

In [22] a middleware service selection approach was proposed titled SSM_EC which utilizes outranking ELECTRE algorithm. QoS from the consumers and providers are collected, based on that information the concordance index, the discordance index and the credibility degrees are calculated. With the concordance index reliability of the outranking relation between two candidates can be presented for a given criterion. For the discordance index, it is used to determine the correctness of the outranking relation based on the performance difference between the two alternatives by using the criterion. But the credibility degree combines both the concordance and discordance index to provide the outranking relation for the whole set of criteria. Finally, ordering of the services ascending and descending is performed and the rank of the services is calculated.

Other approaches [23, 24] used simple additive weight (SAW)-based methods to rank alternatives for cloud service adoption. Based on decision-making theories in [23], the authors analysed problems which might be encountered by consumers criteria during service selection process and identified solution which can be used in the process. For the proposed service selection framework—MADMAC (i.e. Multiple Attribute Decision Methodology for Adoption of Clouds), the author introduced an attribute hierarchy which consisted of six attributes to define criteria for the decision-making process. With the proposed method, experts' opinions were also taken into consideration and used to determine the value of the criteria. Finally, a SAW-based method was used to calculate the rank of the services.

Zhao et al. [25] proposed SPSE (Service Provider Search Engine), for service selection and scheduling. The approach focuses on service selection in SOA and cloud computing environments with the consideration of users' personalization and multiple objectives. The proposed

method includes four basic operations: search, filter, rank, and update. In the first step, by using indexing technology, services with required service attributes and available service providers are searched. The found services are then filtered via a pareto optimal-based selection method to improve scheduling efficiency. In the third step, services are ranked and each parameter of the service is first ranked in light of the values provided by the service provider. The ranked values multiplied with consumer preferences are then added to arrive at the final ranking of the service. The update operation automatically calculates and updates the consumer preferences according to the first requirements of consumers and their subsequent choices of services.

Unfortunately many of the existing approaches use a precise value comparison for QoS parameters. Any missing QoS parameters of a service lead to exclusion of that service in the process of ranking and selection. The suggested ranking techniques should be tolerant with partial matching of parameters. This tolerance, however, should be exercised with care, as it might lead to failure of selection process and consequently fail to satisfy the needs of the service consumer. In contrast to SPSE- [25] and SAW-based [23] approaches, we develop an automatic method to autonomously determine the weight value of QoS attributes by calculating correlation dependencies between different QoS attributes and a new service ranking algorithm is introduced by considering partial matching of services, which will be introduced in the following section.

3 System model

3.1 QoS parameters

As the number of services increases, it becomes inevitable that there will be multiple services with similar functionality performing the same task but with different quality. For clarification, the 'task' describes the process of providing information and the 'quality' describes how the task is satisfied. Quality, when used in this context, refers only to the task and not the actual data returned. QoS in internet of services is usually used to represent non-functional characteristics of services. Different non-functional QoS attributes of services can be divided into two parts, one part as the user-independent properties which have identical values for different users (e.g., price, popularity, etc.), and the second part as the user-dependent properties which have the different values for different service users (e.g., response time, throughput, etc). The QoS values are affected not only by the service providers, but also by the environmental factors in services computing (e.g., server workload, network condition, etc.).

Table 1 Consumer quality of service requirements

Number	QoS requirement	Preferred value
1	Distance	7–32 km
2	Availability	80–90 %
3	Reliability	70–85 %

Researchers have proposed numerous different approaches regarding the use of QoS criteria in services, summaries of many of the methods including benefits and limitations have also been published [26]. It is common in many of the methods to rely only on QoS attribute values published by the service providers. Attributes such as response time and availability may change over time independently of the published values and the values might only be accurate when the service is used within the original network context. All attribute values given by providers are considered subject to validity and integrity.

In the consumer requirements, each QoS parameter can be either mandatory or optional. Optional values are also given a weight factor to determine priority. The tendency of the parameter as shown in Table 1 indicates whether high or low values are regarded as more desirable in an ideal scenario. For example, high availability and throughput values imply a better service, whereas low response time and latency values would also generally indicate better service.

3.2 Architecture

The aim of our proposed system is to provide an efficient service selection tool in which service consumer is able to choose the best available service based on consumer requirements and service QoS attributes. Figure 1 shows the model within more detail.

3.2.1 Consumer request module

The process begins with the consumer supplied requirements, which are different for each consumer and purpose. These requirements initially define the task and desired QoS. As examples, for a given output, a project may prioritize service cost above update rate, whereas another

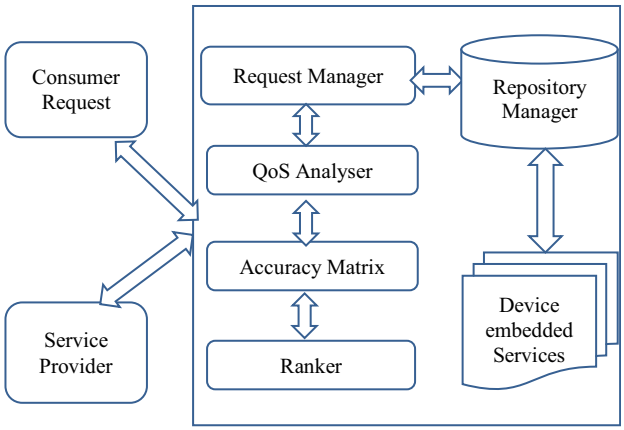

Fig. 1 Service selection model

project might consider higher cost to be justified to obtain a higher update rate.

3.2.2 Request manager module

This module gets the QoS requirements from consumer requirement module and communicates with service repository manager to return the list of available offers which perform consumer required functionality, and extract QoS parameters of the offered services.

3.2.3 QoS analyser module

This module analyses the offers and collects the QoS parameters belong to offers. The parameters will be used in constructing accuracy matrix and calculation of offers ranks.

3.2.4 Accuracy matrix module

From the list of services and their QoS parameters, a matrix will be formed. The matrix will be normalized based on consumer requirements. Within this module, services will be filtered and candidate for the ranking and selection process.

3.2.5 Ranker module

Based on the accuracy matrix, each candidate service will be evaluated and ranked. The detail of the ranking process is discussed in the Ranking Algorithm.

3.2.6 Implementation module

The list of ranked services will be returned to the consumers, and consumers will select the most desired service based on their requirements. In case of failure the implementation, the module will be responsible to find alternative services for the consumer.

3.3 Ranking and recommendation algorithm

This section explains how our algorithm and accuracy matrix are applied to services with only partially available data in order to generate a more complete, preference ranked service listing. For each selection process, there is a list of services which have similar functionalities and may be able to satisfy the task requirement.

Each service can be represented as

$$Service = \{s\_id, s\_name, p\_id, s\_type, I, O, QoS\}$$

where, s_id is identifier for the service, s_name is any given name to the service, p_id is the identifier for the provider of the service, I is a set of service input, O is a set of service output, and QoS is a list of QoS parameters of the service.

Consumer request is used to define QoS requirements,

$$Request = \{request\_id, consumer\_id, i\_data, s\_type, req\}$$

where $request\_id$ is an identifier for the consumer request, $consumer\_id$ is an identifier of the consumer, i_data is a set of data submitted to the provider, s_type defines the type of the service the consumer requires, and req defines list of QoS requirements from the consumer.

Let

- Q_R be a set of consumer QoS requirements, $Q_R = \{r_1, r_2, r_3 \ldots r_n\}$ where n $\in$ N.
- S be a set of potential services with similar functionality, $S = \{s_1, s_2, s_3 \ldots s_m\}$, m $\in$ N.
- Each S candidate services has Q_S property matrices $Q_S = \{Q_{S1}, Q_{S2}, Q_{S3} \ldots Q_{Si}\}$, where $Q_{Si} = \{q_{i1}, q_{i2}, q_{i3} \ldots q_{ij}\}$, $i,j \in N$. Q_{Si} represents quality matrices for service i.

During the process of service selection, its unlikely consumer QoS requirements Q_R has same number of matrices as services QoS parameters Q_{Si}. Q_R is taken as the baseline and quality matrices are arranged as follows,

1. Any consumer Q_R which is lacking in service Q_{Si} will be assigned with 0.
2. Removing any Q_{Si} which is not presented in Q_R.

If n consumer QoS requirements Q_R has been identified, and m potential services can satisfy consumer functional requirements, the m-by-n requirements matrix is constructed, R, as shown in Fig. 2. Each column in the matrix represents consumer QoS requirements Q_R, and each row represents a potential service for the selection process.

Fig. 2 Requirements matrix, R

$$R = \begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ . \\ S_m \end{array} \begin{array}{c} \begin{array}{ccc} Q_{R1} & Q_{R2} \cdots & Q_{Rn} \end{array} \\ \begin{pmatrix} r_{11} & r_{12} \cdots & r_{1n} \\ r_{21} & r_{22} \cdots & r_{2n} \\ r_{31} & r_{32} \cdots & r_{3n} \\ \vdots & \vdots & \vdots \\ r_{m1} & r_{m2} \cdots & r_{mn} \end{pmatrix} \end{array}$$

With the consideration of the original consumer QoS requirements, all services that cannot satisfy mandatory requirements are removed from selection process as shown in Algorithm 1. The remaining candidate services will be considered in the matchmaking process.

The calculation of the accuracy matrix, A, is dependent on the tendency of QoS parameters. Tendency explains how the numeric value of a service changes for the service to be perceived as better. The tendency for the majority of values in our example is expected to be high, only response time and latency are expected to be low.

Using the consumer specified QoS range and the service derived QoS offered, each element of the accuracy matrix, A, is calculated using the case-dependant formulae (1)–(6).

For values with high tendency:

$$\frac{R_{ij}}{R_l} \quad \text{when} \quad R_{ij} < R_l \tag{1}$$

$$\frac{R_{ij} - R_l}{R_h - R_l} + \alpha \tag{2}$$

$$\frac{R_{ij}}{R_{\max}} + \beta \quad \text{when} \quad R_{ij} > R_h \tag{3}$$

For values with low tendency:

$$\frac{R_h}{R_{ij}} \quad \text{when} \quad R_{ij} > R_h \tag{4}$$

$$\frac{R_h - R_{ij}}{R_h - R_l} + \alpha \quad \text{when} \quad R_l \le R_{ij} \le R_h \tag{5}$$

$$\frac{R_{\min}}{R_{ij}} + \beta \quad \text{when} \quad R_{ij} < R_l \tag{6}$$

where R_{ij} is the value of ith QoS property of jth service, R_l is the lower limit of consumer requirement for an attribute. R_h is the higher limit of consumer requirement for an attribute. R_{max} is the maximum value of a QoS property offered by the services under consideration. R_{min} is the minimum value of a QoS property offered by the services under consideration. α and $\beta \in \{1, 2, 3 \ldots\}$ where $\alpha < \beta$.

Algorithm: Partial Service Ranking and Recommendation based on QoS properties

Input: CR (Consumer Requirements), SL (Service List)

For all service s ∈ {1, 2, 3 ... m} in SL **do**

If (s satisfies mandatory consumer requirements)

Add s to CL (Candidate List)

// Candidate List CL is a list of all relevant services

end if

end for

For all service s in CL

For all QoS parameter q

Find Minimum and Maximum value of q from SL

//Min and Max will be used in formation of accuracy matrix.

end for

end for

For all service s in CL

Calculate normalised value of QoS property

// normalized values will be calculated using equation 1-6

end for

For all service s in in CL

Calculate total score for each service s

$$R_s = \sum_{j=1}^{n} A_{is} * W_s$$

end for

Based on the total score Rank all services order services in CL

Return CL

Figure 3 shows three different ranges of interest: preferred, tight and loose. Figure 3a shows increasing of values for high tendency QoS properties, while Fig. 3b shows decreasing of values for low tendency QoS properties.

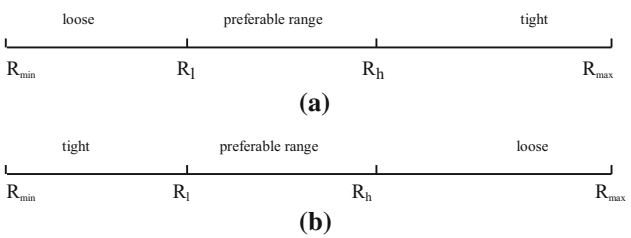

Fig. 3 QoS property measurement **a** high tendency, **b** low tendency

The Eqs. (1)–(6) generate results which are normalized in the range 0–1. The constants α and β are introduced in order to discriminate between the three ranges. For results in the loose range, the value remains between 0 and 1. For results in the preferable range, α is added to the equation and the results are in the range α to $(\alpha + 1)$. For results in the tight range, β is added and the results are in the range β to $(\beta + 1)$.

For consistency, we have normalized values of QoS properties to be in a small range using Eqs. (1)–(6), in which all the values in the accuracy matrix lies in the range of $(0, \beta + 1)$. In the selection process, there are thousands of services, the main purpose of the algorithm is to arrange all the QoS values for services based on minimum and maximum values, and arrange the values between $(0, \beta + 1)$.

On the other hand, every value in the algorithm is considered precisely based on the range. Considering a value for availability; lets assume preferable range is (80–90 %). Any service having availability value less than 80 % will be counted as loose, those between 80 and 90 % will be counted as preferable range and α will be added, finally any value larger than 90 % will be counted as tight and β will be added. Since $0 < \alpha < \beta$, it guarantees that the higher range always has higher value than the other two ranges.

The results of the calculations are used to populate the accuracy matrix, A. The matrix shows how accurately each advertised service matches the overall consumer requirement without yet considering the desired preference of each attribute.

3.4 Weight calculation

Collecting weight values from service consumers and calculating weight values using arithmetic and geometric methods might not be an efficient or understandable choice. They are not the most suitable choice for complex selection process with tens or hundreds of consumer requirements with higher weights. Dependency between parties is always considered important. QoS properties have different values and there are dependencies between these parties. Values in QoS properties might affect each other, which will directly affect performance and accuracy of the service selection process. The paper considers these dependencies to determine weighting value of QoS properties using distance correlation between QoS properties. Distance correlation can detect different types of dependencies, the value will be one when the two QoS properties are totally dependent on each other, and zero when the two QoS properties are statistically independent.

The distance correlation for a sample $(X,Y) = \{(X_k, Y_k): k = 1 \ldots p\}$ from the pairwise distribution of random vectors X and Y defined as follows:

$$a_{jk} = ||X_j - X_k||, \quad b_{jk} = ||Y_j - Y_k|| \quad j,k = 1,2,\ldots,p$$

It is worth to note that $\|_{\mathrm{p}}$ denotes the Euclidean norm, similarly,

$$A_{j,k} := a_{j,k} - \bar{a}_{j.} - \bar{a}_{.k} - \bar{a}_{..}, \quad B_{j,k} := b_{j,k} - \bar{b}_{j.} - \bar{b}_{.k} - \bar{b}_{..},$$

where $\bar{a}_{j.}$ is the mean of jth row, and $\bar{a}_{.k}$ represents kth column mean, and $\bar{a}_{..}$ represents grand mean of the distance matrix of the X, it is the same for the b values.

The distance covariance of the two distributions X and Y is computed as follows:

$$\mathrm{dCov}_p^2(X,Y) := \frac{1}{p^2}\sum_{j,k=1}^{p} A_{j,k} \times B_{j,k} \tag{7}$$

Finally, with the equation below, the distance correlation between the two distributions X and Y is calculated as following:

$$\mathrm{dCor}(X,Y) = \frac{\mathrm{dCov}(X,Y)}{\sqrt{\mathrm{dCov}(X) \times \mathrm{dCov}(Y)}} \tag{8}$$

With computing pairwise distance correlations for different QoS attributes, it produces the following dependency matrix, the overall dependency of QoS_i with respect to other quality attributes is determined as follows:

$$\mathrm{dep}(\mathrm{QoS}_i) = \frac{1}{p}\sum_{j=1}^{p} \mathrm{dCor}_{ij} \tag{9}$$

The weight of each quality attributes is then derived from the dependency value as follows:

$$W(\mathrm{QoS}_i) = \frac{\mathrm{dep}(\mathrm{QoS}_i)}{\sum_{i=1}^{m} \mathrm{dep}(\mathrm{QoS}_i)} \tag{10}$$

After the A matrix is fully populated, the rank of each service is calculated by summation of the QoS attribute and weight product for that service using (11).

$$R_i = \sum_{j=1}^{n} A_{ij} \times W_j \tag{11}$$

where R_i represents rank of service i, A_{ij} represents the accuracy value of jth QoS property of service i, and W_j represents weight of jth QoS property.

Example We are taking a real-world example and going through it step by step to clearly demonstrate the service ranking algorithm.

The plan is to use emergency support service. For the sake of simplicity, we assume there are three QoS requirements as in Table 1.

Assume the list of available emergency support services available are as below in Table 2,

By using Eqs. (1)–(6) we can calculate QoS property values for each service as shown in Table 3. According to the emergency case preferred values, S_1 does not comply

Table 2 List of available services

No.	Service name	Distance (km)	Availability (%)	Reliability (%)
S_1	Emergency support	4.93	42	73
S_2	Plural emergency	64.5	86	85
S_3	Fast emergency	10.3	85	90
S_4	Global emergency	28.5	90	70
S_5	County emergency	50	97	73

Table 3 Values of QoS for each candidate service

No.	Service name	Distance (km)	Availability (%)	Reliability (%)
S_1	Emergency support	3	0.525	1.2
S_2	Plural emergency	0.496	1.6	2
S_3	Fast emergency	1.868	1.5	3
S_4	Global emergency	1.14	2	1
S_5	County emergency	0.64	3	1.2

Table 4 Rank of each service

Services	Final score
S_1	18.975
S_2	11.28
S_3	19.84
S_4	13.7
S_5	14.6

with all QoS requirements, S_1 value for availability is 42 % which is lower than the preferred range, and this is true for both S_2 and S_5. Since none of the QoS requirements is mandatory, they can be considered in the selection process.

Based on the results from Table 3, scores of each service can be calculated by using Eq. (11). It is considered the corresponding weight value is given for the request, and the weight value for the three QoS requirements are assumed to be {5, 3, 2}, respectively.

From the Table 4, it can be understood that services returned to the consumer are ranked as $S_3 > S_1 > S_5 > S_4 > S_2$. Service S_3 is having a distance of 10.3 km with availability of 86 % and reliability of 90 %. While S_1 has value of 4.93 km for distance which is better than distance value of S_3, but S_1 availability value is 42 % and reliability is 73 %. Based on our approach by considering the overall criteria, S_3 is the best suited to the consumer request.

4 Experimental results and evaluations

4.1 Experiment setup

In this section, experimental results are shown for the algorithm. For the experiment, a large-scale dataset, WS_DREAM, has been used. The WS-DREAM dataset three [27] is a large-scale dataset which has more than 4500 real-world services. Each service was invoked by 142 service consumers in 64 different time intervals. Each service has different QoS properties like response time and throughput. Based on the QoS properties, service ratings are calculated by using multi-attribute utility function [28].

The experiments were carried out on a HP ProBook laptop Core i5 2.50 GHz, with 4 GB of RAM on Windows 7 Enterprise. In the evaluation, a set of experiments were conducted. Experiments are to evaluate the algorithm based on accuracy and performance.

4.2 Performance measurement

In the proposed algorithm, the rank of services was calculated based on QoS properties. To measure the performance, the algorithm was compared to the algorithm in [24] which used the probabilistic flooding-based method, by combining simple additive weighting (SAW) technique and Skyline filtering. In addition, it was compared to SPSE algorithm, which uses the pareto optimal-based solution method [25]. In the experiment, a real-world services were used.

The discounted cumulative gain (DCG) method was used, which is a common method for information retrieval and quality ranking [29]. The method calculates the quality of services depending on their rank in the list; the gain is accumulated at the upper ranks and discounted in the lower ranks. Services with better quality should be shown earlier in the ranking method.

The equation is as follows,

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{Q_i} - 1}{\log(1 + R_i)} \tag{12}$$

where Q_i is the QoS for the ith service, and R_i is the rank of ith service. The higher the $\mathrm{DCG_P}$ value, the higher the QoS

Table 5 DCG ranking for accuracy matrix, SPSE, and SAW

Top P	Acc. matrix	SPSE	SAW
5	7.569091655	4.646180572	3.0730903
10	10.38841135	6.38846729	3.9442336
15	12.71343601	8.372440175	4.9362201
20	14.89674501	10.34730971	5.9236549
25	16.85575592	12.17659888	6.8382994
30	18.62296682	14.09352166	8.1967608
35	20.40988547	15.89264909	10.496325
40	21.8047093	17.3287577	11.864379
45	23.0047093	19.6287577	13.314379
50	24.8047093	22.1287577	16.514379

parameters of the Top-P service. The experiment compared the ranking value of the top 50 services and is summarized in increments of 10.

Table 5 shows the results of DCG for top-50 services returned for the selection process. From the results, we can see the quality matrices of accuracy matrix top-P services are higher and in many cases, nearly double to SPSE and SAW results. For example in the analysis category, the top 5 service for accuracy matrix algorithm is 7.569, while for SPSE is 4.646, and for SAW is 3.073. It is clear that the top 5 services are the most important services to the consumer, since most of the times consumer selects services from the top 5 candidates. By examining the returned results, it can be understood that the proposed algorithm returns higher quality services to the consumers and provides more satisfactory results to their requirements and this is due to the high accuracy of our approach in which each QoS property plays important role in the ranking and recommendation process.

4.3 Precision and recall

The precision and recall are standard measurements that have been used in information retrieval for measuring the accuracy of a discovery or recommendation method or a discovery engine. The precision is defined as the ratio of the number of returned correct services to the total number of all returned services [30]. By contrast, recall is defined as the ratio of the number of returned correct serviced to the number of all correct services as shown in Eq. (12).

$$\text{Prec} = \frac{\text{Rtrel}}{\text{Rel}}, \quad \text{Rcal} = \frac{\text{Rtrel}}{\text{Rt}} \tag{13}$$

where Prec represents precision, Rtrel represents a list of returned relevant services, Rel represents relevant services, and Rt represents returned services.

To assess the method, we categorized the services into different categories based on the functionalities they provide. From one category, we randomly selected a set of 60 services with different QoS properties, of which 40 services were relevant to the query. We conducted 10 different tests and each of them repeated 100 times. In the first test no service had unmatched properties. In the second test, at least one of the properties was partial matched to the service query. The numbers of partial match properties were increased through remaining tests.

In Fig. 4, it can be depicted that the precision is affected by the number of returned services and it decreases as the number of returned services increase. On the other side, the recall increases by increasing number of returned services as shown in Fig. 5. From Figs. 4 and 5, it is apparent that the accuracy matrix algorithm improved both precision and recall, the percentage of relevant services in the accuracy matrix is higher than SPSE and SAW. The improvement is due to the inclusion of partially matched services in the

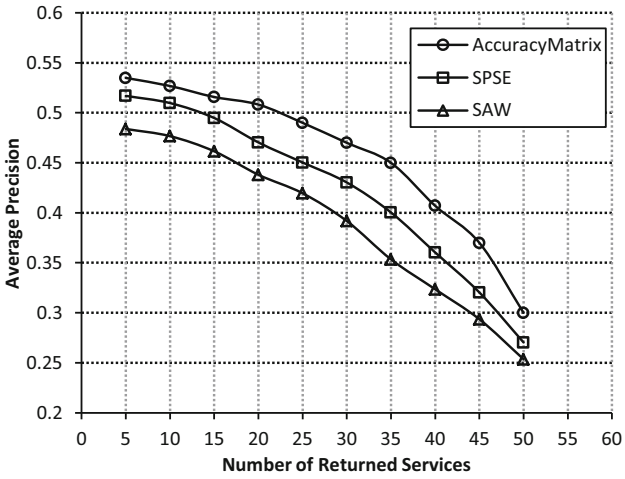

Fig. 4 Comparison of precision between SPSE and Acc. matrix

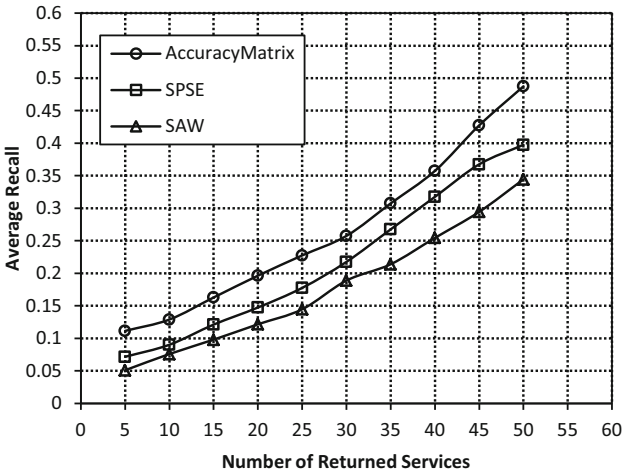

Fig. 5 Comparison of recall between SPSE and Acc. matrix

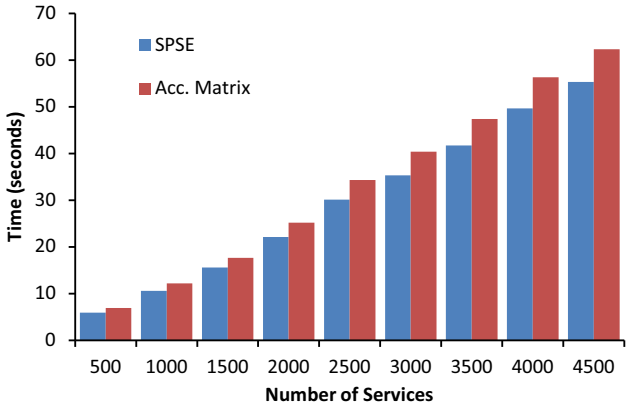

Fig. 6 Performance evaluation with fixed number of QoS parameters

accuracy matrix algorithm. This effect is very clearly observed when the recall for SPSE and SAW do not reach as high as our algorithm; it is because of the limitation of service matchmaking in both algorithms by exclusion of partially matched services. This confirms that our approach deals more accurately with the service query and returns relevant service to the consumer even in the case where the candidate services only partially match the consumer requirements.

4.4 CPU time

In this part of the experiment, we evaluate the time consumption of our experiments with increasing number of services which will reflect the computational cost and scalability of the evaluated methods. A real-world dataset were used, which had up to 4500 services. The consumed time was measured while increasing the number of services.

From results in Fig. 6, the consumed time increases linearly with an increasing number of services, and the consumed time of our method is comparable with SPSE. But our method achieved a better trade-off in term of performance and overhead. For example, with 1000 services, our algorithm returned 56 services in 12.2 s, while SPSE returned 36 services in 10.6 s. The figure also shows the direct proportionality between execution time and services considered. The slope of this linearity strongly implies that our approach is rather scalable with increasing number of services.

5 Conclusion and future work

In this work, we proposed an efficient algorithm to service selection in the IoT environment. By considering all QoS properties, the algorithm includes consumer preference and allows the consumer to select the best available service for their task. The design of an accurate ranking algorithm for services based on matching consumer requirements identifies those services which are candidates for use in the selection process.

When an initial list of services is returned which only partially matches with optional consumer requirements, our algorithm includes them in the decision and recommendation process. This holds true even in the case where the optional attributes barely match. This action allows more services to be included and accurately ranked, providing the best choice to the consumer. Experimental results show that in extreme circumstances, our algorithm recommends services when other methods would return an empty list. This feature can be used and applied to other service applications such as service discovery and composition.

The solution is generic and can deal with multiple QoS properties and large numbers of services. The algorithm is robust, scalable and efficient. Experimental results show a significant improvement in the number of relevant services recommended by our algorithm when compared to the SPSE algorithm. The scalability was tested by increasing number of services and QoS parameters. There was no significant performance degradation when reducing large datasets with multiple QoS attributes. Our experiments rendered results that strongly support its use as a tool for service selection and recommendation processing.

In future work, we will focus on some other aspects of service selection process. For example, concentrating on further refinery of our selection and recommendation algorithm results by the inclusion of consumer opinion, which will help in the prediction of the user QoS requirements and assigned weight value.

Acknowledgments The work reported in this paper has been supported by National Natural Science of Foundation of China Program (61502209 and 61502207), Natural Science Foundation of Jiangsu Province of China (BK20130528) and Visiting Research Fellow Program of Tongji University (8105142504).

References

1. Guinard D, Trifa V, Karnouskos S, Spiess P, Savio D (2010) Interacting with the SOA-based internet of things: discovery, query, selection, and on-demand provisioning of web services. IEEE Trans Serv comput 3(3):223–235

2. Yan L, Zhang Y, Yang LT, Ning H (2008) The internet of things: from RFID to the next-generation pervasive networked systems (Hardback). Auerbach Publications, New York
3. Health and Social Care Information Centre (2014). http://www.hscic.gov.uk/catalogue/PUB13464/acci-emer-atte-eng-2012-2013-rep.pdf
4. Rao J, Su X (2004) A survey of automated web service composition methods. In Proceedings of the first international workshop on semantic web services and web process composition, San Diego, California, USA, pp 43–54
5. Wu Y, Yan C, Liu L, Ding Z, Jiang C (2015) An adaptive multilevel indexing method for disaster service discovery. IEEE Trans Comput 64(9):2447–2459
6. Chang C, Srirama S, Mass J (2015) A middleware for discovering proximity-based service-oriented industrial internet of things. In 2015 IEEE international conference on services computing (SCC), pp 130–137
7. Lindenberg J, Pasman W, Kranenborg K, Stegeman J, Neerincx M (2006) Improving service matching and selection in ubiquitous computing environments: a user study. Pers Ubiquitous Comput 11(1):59–68
8. Agarwal S, Studer R (2006) Automatic matchmaking of web services. In: Proceedings of the IEEE international conference on web services, pp 45–54
9. Liu L, Antonopoulos N, Zheng M, Zhan Y, Ding Y (2016) A socio-ecological model for advanced service discovery in machine-to-machine communication networks. ACM Trans Embed Comput 15:38
10. Plebani P, Pernici B (2009) URBE: web service retrieval based on similarity evaluation. IEEE Trans Knowl Data Eng 21(11):1629–1642
11. Klusch M, Fries B, Sycara K (2009) OWLS-MX: a hybrid semantic web service matchmaker for OWL-S services. Web Seman Sci Serv Agents World Wide Web 7(2):121–133
12. Chao K, Younas M, Lo C, Tan T (2005) Fuzzy matchmaking for web services. In: 19th international conference on advanced information networking and applications, pp 721–726
13. Huang C, Chao K, Lo C (2005) A moderated fuzzy matchmaking for Web services. In The fifth international conference on computer and information technology, pp 1116–1122
14. Ngan L, Kanagasabai R (2013) Semantic Web service discovery: state-of-the-art and research challenges. Pers Ubiquitous Comput 17(8):1741–1752
15. Kritikos K, Plexousakis D (2014) Novel optimal and scalable nonfunctional service matchmaking techniques. IEEE Trans Serv Comput 7(4):614–627
16. Liu Y, Ngu A, Zeng L (2004) QoS computation and policing in dynamic web service selection. In: Proceeding 13th international conference World Wide Web, pp 66–73
17. Cezar Estrella J, Santana RH, Santana M, Bruschi S (2012) WSARCH: a service-oriented architecture with QoS, chap 16. In: Handbook of research on service-oriented systems and non-functional properties: future directions, IGI Global, pp 352–380
18. Kritikos K, Plexousakis D (2012) Towards optimal and scalable non-functional service matchmaking techniques. In IEEE 19th international conference on web services (ICWS), pp 327–335
19. Pernici B, Siadat S (2011) Selection of service adaptation strategies based on Fuzzy logic. In IEEE World Congress on Services (SERVICES), pp 99–106
20. AlMulla M, Almatori K, Yahyaoui H (2011) A QoS-based Fuzzy model for ranking real world web services. In: IEEE international conference on web services (ICWS), pp 203–210
21. Pernici B, Siadat S (2011) A fuzzy service adaptation based on QoS satisfaction. In: Proceeding of the 23rd international conference on advanced information systems engineering, London, UK, pp 20–24
22. Silas S, Rajsingh EB, Ezra K (2012) Efficient service selection middleware using ELECTRE methodology for cloud environments. Inf Technol J 11(7):868–875
23. Saripalli P, Pingali G (2011) MADMAC: multiple attribute decision methodology for adoption of clouds. In IEEE international conference on cloud computing, pp 316–323
24. Lin W, Dou W, Xu Z, Chen J (2013) A QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network. Concur Comput 25(13):1843–1860
25. Zhao L, Ren Y, Li M, Sakurai K (2012) Flexible service selection with user-specific QoS support in service-oriented architecture. Netw Comput Appl 35(3):962–973
26. Platenius M, Detten M, Becker S, Schäfer W, Engels G (2013) A survey of fuzzy service matching approaches in the context of on-the-fly computing. In CBSE '13 Proceedings of the 16th international ACM Sigsoft symposium on component-based software engineering, NY, USA, pp 143–152
27. Zhang Y, Zheng Z, Lyu MR (2011) WSPred: a time-aware personalized QoS prediction framework for web services. In: Software reliability engineering (ISSRE), pp 210–219
28. Zeng L, Benatallah B, Dumas M, Kalagnanam J, Sheng QZ (2003) Quality driven web services composition. In: In proceedings of the 12th international conference on World Wide Web (WWW '03), New York, pp 411–421
29. Kekalainen K, Jarvelin J (2002) Cumulated gain-based evaluation of IR techniques. ACM Trans Inf Syst 20:422–446
30. Skoutas D, Sacharidis D, Kantere V, Sellis T (2008) Efficient semantic web service discovery in centralized and P2P environments. In: Proceedings of the 7th international semantic web conference(ISWC 2008). Springer, Berlin, pp 583–598